Multi-scale CAFE framework for simulating fracture in heterogeneous materials implemented in Fortran coarrays and MPI

Anton Shterenlikht
Mech Eng Dept
The University of Bristol
Bristol BS8 1TR, UK
Email: mexas@bris.ac.uk

Lee Margetts
and Jose D. Arregui-Mena
School of Mechanical,
Aero and Civil Engineering
The University of Manchester
Manchester M13 9PL, UK
Emails: Lee.Margetts@manchester.ac.uk
jose.arregui-mena@manchester.ac.uk

Luis Cebamanos
Edinburgh Parallel Computing Centre (EPCC)
The University of Edinburgh, King's Buildings
Edinburgh EH9 3FD, UK
Email: l.cebamanos@epcc.ed.ac.uk

***Abstract*—Fortran coarrays have been used as an extension to the standard for over 20 years, mostly on Cray systems. Their appeal to users increased substantially when they were standardised in 2010. In this work we show that coarrays offer simple and intuitive data structures for 3D cellular automata (CA) modelling of material microstructures. We show how coarrays can be used together with an MPI finite element (FE) library to create a two-way concurrent hierarchical and scalable multi-scale CAFE deformation and fracture framework. Design of a coarray cellular automata microstructure evolution library CGPACK is described. A highly portable MPI FE library ParaFEM was used in this work. We show that independently CGPACK and ParaFEM programs can scale up well into tens of thousands of cores. Strong scaling of a hybrid ParaFEM/CGPACK MPI/coarray multi-scale framework was measured on an important solid mechanics practical example of a fracture of a steel round bar under tension. That program did not scale beyond 7 thousand cores. Excessive synchronisation might be one contributing factor to relatively poor scaling. Therefore we conclude with a comparative analysis of synchronisation requirements in MPI and coarray programs. Specific challenges of synchronising a coarray library are discussed.**

***Index Terms*—Computer aided engineering, Scientific computing, Parallel programming, Supercomputers, Parallel algorithms, Mechanical engineering, Microstructure**

1. Introduction

Many deformation and fracture problems of solid mechanics involve multiple competing physical processes occurring at different time and length scales. A variety of multi-scale modelling approaches have been proposed to treat such problems, e.g. combined atomistic and continuum mechanics [1], molecular dynamics and continuum mechanics [2], discrete dislocation and continuum plasticity [3], etc. The cellular automata (CA) method has been used together with finite elements (FE), in a multi-scale CAFE approach for problems involving material microstructure, such as solidification [4], [5], recrystallisation [6] or fracture of polycrystals [7]–[10]. FE is used to solve the continuum mechanics problem (coarse scale) to calculate the macroscopic quantities, such as the strain, stress or temperature gradients, while the microstructure (fine scale) is updated with the CA method. Each iteration of the CAFE model continuum mechanics quantities are passed from the coarse FE scale to the fine CA scale (localisation) and damage variables are passed from the CA scale back to the FE scale (homogenisation) [11]. Thus CAFE is a two-way hierarchical concurrent multi-scale framework [12].

In the 3D CAFE method the space is partitioned into identical cells, e.g. cubic. Cells have physically meaningful states, e.g. liquid phase, crystal with a certain rotation tensor, crack front, crack flank, cleavage plane of a particular type, etc. The state of each cell at the next iteration is determined by the state of that cell, the states of its immediate neighbourhood cells, (e.g. the 26-cell Moore's neighbourhood) and some continuum FE field variables (e.g. stress, strain or temperature), all taken at the current iteration. The fact that the CA method has an explicitly "local" domain of influence, with no global equilibrium requirements, opens opportunities for parallelisation. Each cell can be updated independently - in parallel. The cell update algorithm is much simpler than in the FE method. Hence a much higher CA resolution can be achieved compared to the FE method, for the same computational cost.

CA spaces can be infinite or finite. A finite CA space is most useful if used with regular boundaries, typically periodic (self-similar) or fixed. Coarrays, a Fortran native means for SPMD programming [13]–[15], are a natural implementation choice for CA models. A 3D CA space with cubic cells of discrete states maps perfectly onto a 3D integer array coarray. In contrast the FE part of the CAFE model, which implements the coarse scale continuum solid

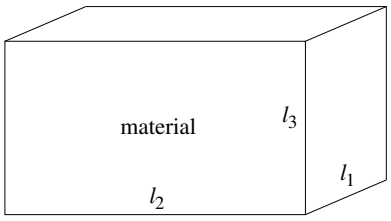

Figure 1. Schematic of the CA space.

mechanics, typically has irregular boundaries. Most often MPI is used to implement parallel Lagrangian FE solvers. In this work a hybrid coarray/MPI parallel paradigm is used to implement a flexible, expandable and scalable CAFE framework.

Coarrays can coexist with other parallel technologies, such as MPI or OpenMP, although to date there only a few examples of such hybrid codes. The European Centre for Medium-range Weather Forecasts (ECMWF) have used coarrays in combination with MPI and OpenMP in their codes [16]. Coarrays have been used together with OpenMP in plasma codes [17].

In the following sections we describe how the CAFE model was implemented using a hybrid coarray/MPI approach. We present strong scaling results and discuss synchronisation challenges.

2. The coarray/MPI framework

ParaFEM, a highly scalable and portable MPI FE library written in Fortran 2003, was used in this work, `parafem.org.uk`, [18], [19]. Previously ParaFEM has been used in large scale simulations in nuclear fusion research [20], [21] and bio-mechanics [22], [23].

CGPACK is a scalable CA library written in Fortran 2008 with extensive use of coarrays, `cgpack.sf.net`. Work on CGPACK started in 2013 on HECToR, then the UK national HPC system [24]. CGPACK has since been ported to Intel and OpenCoarray/GCC platforms [25], [26].

Both ParaFEM and CGPACK are being actively developed, including contributions from the UK Software Sustainability Institute, `software.ac.uk`, and grants from the embedded CSE programme of ARCHER, the current UK national HPC system. Both ParaFEM and CGPACK libraries are distributed under BSD license.

2.1. Size of the CA coarray

CGPACK module `cgca_m2phys` deals with physical units and sizing of the main CA coarray.

The 3D CA space is used to represent a rectilinear volume of material microstructure, of physical dimensions $l_1 \times l_2 \times l_3$, see Fig. 1. The CA space is implemented as a 4D integer allocatable array coarray, with a 3D coindex set. The first 3 array dimensions identify a particular CA cell. The fourth array dimension is used to store multiple types of microstructural information, e.g. grains or damage [27]:

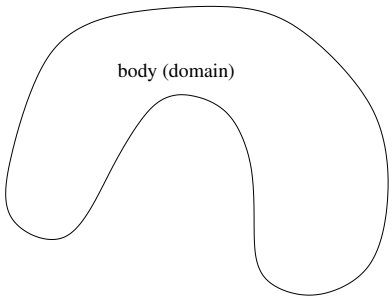

Figure 2. Schematic of the FE domain.

```
integer,allocatable :: space(:,:,:,:)[:,:,:]
```

The exact dimensions and codimensions of the coarray `space` are chosen at runtime, based on the available number of images, N. First the codimensions are chosen, c_1, c_2, c_3, such that $c_1 \times c_2 \times c_3 = N$. Arbitrarily we set $c_1 \geq c_2 \geq c_3$. The codimensions are chosen to minimise $c_1 - c_3$, i.e. to make the coarray grid as 'cubic' as possible. This is advantageous because it minimises the total number of halo cells, and thus the amount of remote data to transfer. The quality of partitioning the microstructure into a 3D array of images is assessed by $q = 1 - (c_1 - c_3)/(N - 1)$, so that $q = 1$ means $c_1 = c_3$, i.e. the lowest possible number of halo cells while $q = 0$ means that $c_1 = N, c_2 = c_3 = 1$, indicating that the number of halo cells is maximised.

Prior work showed that mesh independent CA results are achieved when each crystal (grain) is represented by at least 10^5 cells on average [28]. Then, given the desired microstructure mean grain size, $\overline{d}$, the first 3 dimensions of `space` are calculated.

As an example consider a simulation of a $12 \times 12 \times 20$ mm volume of polycrystalline microstructure with $\overline{d} = 2$mm on 192 images. Array `space` with 2 types of microstructural information is then allocated as:

```
allocate( space(35,70,77,2) [8,4,*] )
```

where $c_3 = 6$. This allows for simulating 360 grains with $q = 0.98$, with the linear resolution of 23.2 cells per mm. The total size of the CA model is $280 \times 280 \times 462 \approx 36$ million cells. In general it is not possible to represent physical space with the exact given dimensions, with the same linear resolution along each coordinate axis, as a discrete CA space. In this example, the volume of microstructure that is actually simulated is $12.06 \times 12.06 \times 19.91$ mm.

2.2. Establishing the CA to FE mapping

CGPACK module `cgca_m3pfem` contains data structures and subroutines which establish a mapping between the CA space and the FE mesh. A schematic example of an irregular FE domain is shown in Fig. 2. Sometimes, the CA space will be fully inside the FE model, but in general, the CA space can be of arbitrary size and orientation with respect to the FE domain, depending on what deformation and/or fracture phenomena are to be studied with it, as

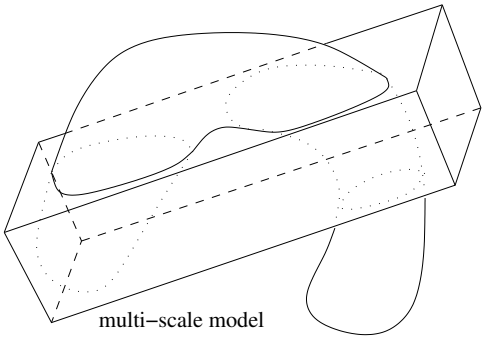

Figure 3. Schematic of a multi-scale CAFE model composed of the FE domain superimposed with the CA material space.

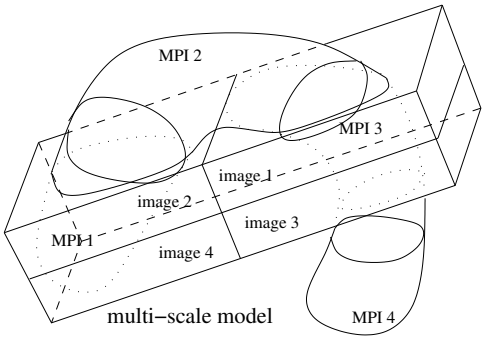

Figure 4. Possible partition of the multi-scale model on 4 PEs.

shown in Fig. 3. Some FEs will occupy the same physical space as some CA cells. These FEs and cells form a two-way macro/micro multi-scale CAFE model. However, as indicated in Fig. 3, in general, there will be cells occupying physical space outside of the body. Such cells do not participate in a multi-scale CAFE analysis.

The coarray/MPI CAFE framework is built with an assumption that at runtime there is always an identical number of MPI processes and coarray images, and that the first MPI process and the first image exist on the first processing element (PE), and so on.

A schematic partition of the CAFE model on 4 PEs is shown in Fig. 4. The boxes show on which PE the corresponding parts of the model are stored. For example, "image 1" and "MPI 1" parts of the model are stored on PE 1. However, these FEs do not share physical space with these CA cells. Instead cells on image 1 share physical space with FEs on PE 3, labelled "MPI 3". This is important because information transfer is required only between CA and FE which occupy the same physical space. In this example the MPI part of the model stored on PE 3 will have to communicate with the coarrays stored on PEs 1 and 3.

Communications between the MPI (FE) and the coarray (CA) parts of the coarray/MPI (CAFE) hybrid model are shown schematically with arrows in Fig. 5. The imbalance in the communication pattern is clear. The FE part of the model stored on PE 4 will not communicate with CA at all. However, the FE part of the model stored on PE 1 will need

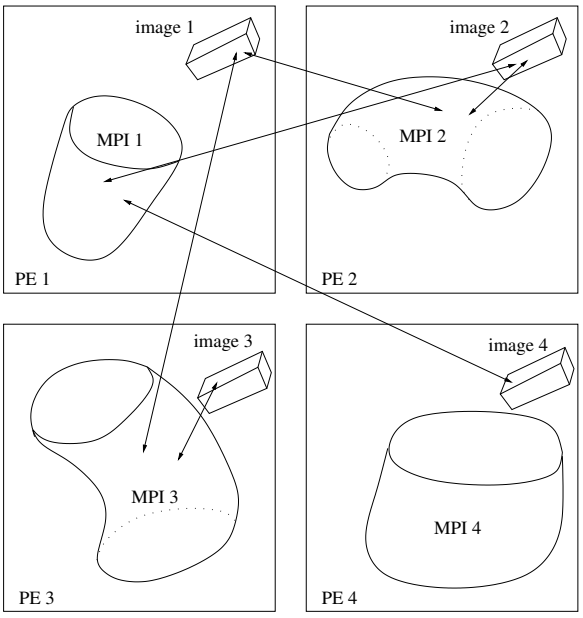

Figure 5. Schematic of communications between the MPI (FE) and the coarray (CA) parts of the coarray/MPI (CAFE) hybrid model on 4 PEs.

to communicate with CA coarrays stored on PEs 2 and 4.

The mapping of FE to CA is established via a private allocatable array of derived type:

```
type mcen
  integer :: image
  integer :: elnum
  real :: centr(3)
end type mcen
type( mcen ), allocatable :: lcentr(:)
```

based on coordinates of FE centroids calculated by each MPI process (`lcentr` stands for local, i.e. non-coarray array of centroids). These coordinates are stored in a coarray of derived type with allocatable array component:

```
type rca
  real, allocatable :: r(:,:)
end type rca
type( rca ) :: centroid_tmp[*]
```

which is allocated as

```
allocate( centroid_tmp%r(3, nels_pp) )
```

where `nels_pp` is the number of FE stored on this PE.

There are two different routines which establish `lcentr` on each image from `centroid_tmp`. Subroutine `cgca_pfem_cenc` implements an all-to-all communication pattern, i.e. each image reads `centroid_tmp` from every image. Subroutine `cgca_pfem_map` uses temporary arrays and coarray collectives `CO_SUM` and `CO_MAX`, which are described in TS18508 [14]. Coarray collectives will be standardised in the next revision of the standard, Fortran 2015. At the time of writing coarray collectives are supported by Cray and OpenCoarray compilers. The

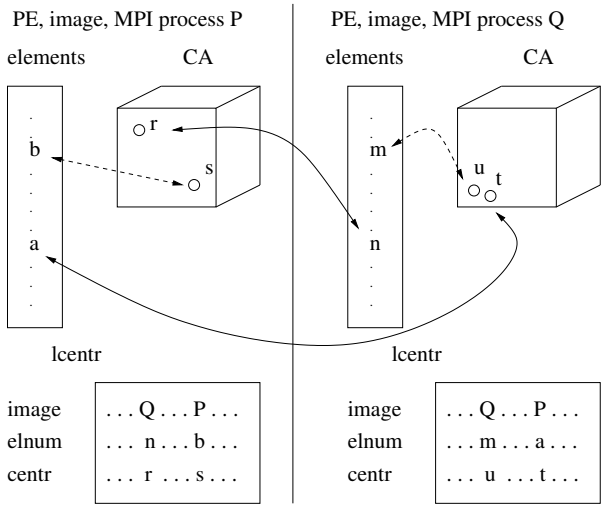

Figure 6. `lcentr` arrays on two images P and Q.

two routines differ in their use of remote communications. However, both routines implement the same algorithm for establishing `lcentr` - if the centroid of an FE on any image is within the CA coarray on this image, then this FE is added to `lcentr` on this image.

Fig. 6 schematically shows `lcentr` arrays established on two images P and Q. In this example finite element n, stored on image Q, has centroid coordinates r, which identify a physical location within the CA coarray on image P. So this element is stored in the `lcentr` array on image P. Finite element m, also stored on image Q, has centroid coordinates u, which identify a physical location within the CA coarray also on image Q. So this element is stored in the `lcentr` array on image Q. FEs with centroids outside of the CA space are not entered in `lcentr`. `lcentr` plays a key role in information transfer between the FE and the CA parts of the multi-scale CAFE model.

2.3. CA cells outside the FE model

After `lcentr` has been established, the second important mapping issue can be resolved. CA cells which are outside of the FE model must not be processed. This means no fracture propagation can occur in such cells. However, because of finite resolution in the FE model and in the CA, this problem cannot be posed precisely. Depending on the FE element size and the CA cell size, a cell can be deemed to lie inside the FE model or outside. The algorithm implemented in the ParaFEM/CGPACK interface uses some characteristic distance measure, L_c. The criterion is this - if the distance between a cell and the centroid of any FE in `lcentr` is less than L_c, then this cell is considered to lie inside the FE model, otherwise it is considered to lie outside of the FE model. Cells which lie outside of the FE model are set to `cgca_state_null` and are not processed in any of the fracture routines. Although these cells represent microstructure in the material layer, this microstructure is simply ignored in all fracture calculations.

This mapping is established with a divide and conquer approach. The algorithm starts by checking boxes of CA cells the size of the whole coarray on each image. If a box is partially in and partially out, it is split into two smaller boxes and the process continues until each box is either fully in, or fully out. If necessary, CA boxes are divided down to single CA cells. This algorithm is implemented in routines `cgca_pfem_partin`, `cgca_pfem_boxin` and `cgca_pfem_cellin`.

2.4. CAFE fracture modelling

Diverse CAFE fracture models can be constructed from the CGPACK and the ParaFEM libraries. The simplest case, presented here, uses a combination of linear isotropic elastic FE with cleavage (fully brittle transgranular fracture mode) CA. Cleavage is the dominant low temperature fracture mode in body centre cubic (bcc) crystals, such as iron. Each time or strain increment of the FE solver the stress tensor, $\mathbf{t}$, is passed to the CA, where it is resolved into normal stresses on $\{100\}$ and $\{110\}$ crystal planes - t_{100}, t_{110} [7], [11]. The localisation (or scatter) algorithm distributes the FE quantities over CA cells based on existing damage in the microstructure, while preserving the FE energy [12].

The cleavage model includes 2 parameters - a fracture stress, σ_F, linked to the free surface energy, γ, and a characteristic length, L. If $t_{100} \geq \sigma_F$ or $t_{110} \geq \sigma_F$, then a CA cleavage crack is extended by L per unit of time. Crack morphology is reduced to a single damage variable, d, by the homogenisation (or gather) algorithm, and the Young's modulus of each FE integration point is reduced according to d, where $d = 1$ means no damage, and $d = 0$ means that the integration point has no load bearing capacity. To avoid numerical instability the FE stiffness is not reduced to below 10^{-3} of the original value (corresponding to $d = 10^{-3}$).

CAFE cleavage simulation is shown in Figs. 7-10. The FE model is a 140mm long mild ferritic steel cylinder of 10mm diameter and 100mm gauge length. One end of the cylinder is constrained and an axial force is applied to the other end. The FE elastic properties are the Young's modulus of 200GPa and the Poisson's ratio of 0.3. Details of the CA material block were given at the end of Sec. 2.1. The CA block is positioned centrally on the cylinder, see Figs. 7-10.

Fig. 7 shows the polycrystalline microstructure layer of the `space` coarray. The colour of each grain (single crystal) encodes its rotation tensor. Fig. 8 shows the grain boundaries in the fracture layer of the `space` coarray. In this model the inactive cells, i.e. cells of state `cgca_state_null` (Sec. 2.3) are not shown. Note that some microstructure still protrudes a little outside the FE cylinder due to a coarse grain resolution of routine `cgca_pfem_partin`, Sec. 2.3.

Fig. 9 shows the macro-crack emerging from linking cracks on preferential cleavage planes in individual crystals. There are 4 cell fracture states in this model: -1, -2, -3 and -4. -1 (yellow) denotes crack flanks on 100 planes. -3 (light blue) denotes crack flanks on 110 planes. Both yellow and light blue regions are clearly visible in Fig. 9. -2 (dark blue)

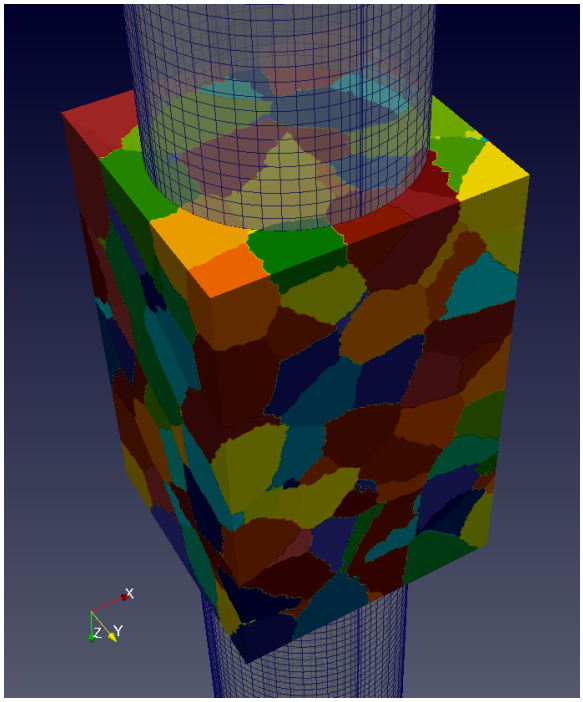

Figure 7. CAFE modelling of a steel cylinder under tension showing the CA microstructure. The FE cylinder mesh is semi-transparent for clarity.

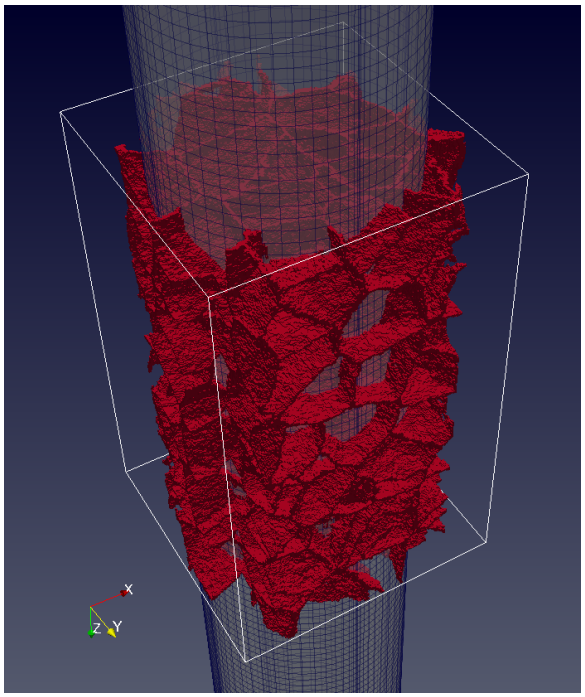

Figure 8. CA microstructure grain boundaries, with inactive cells removed.

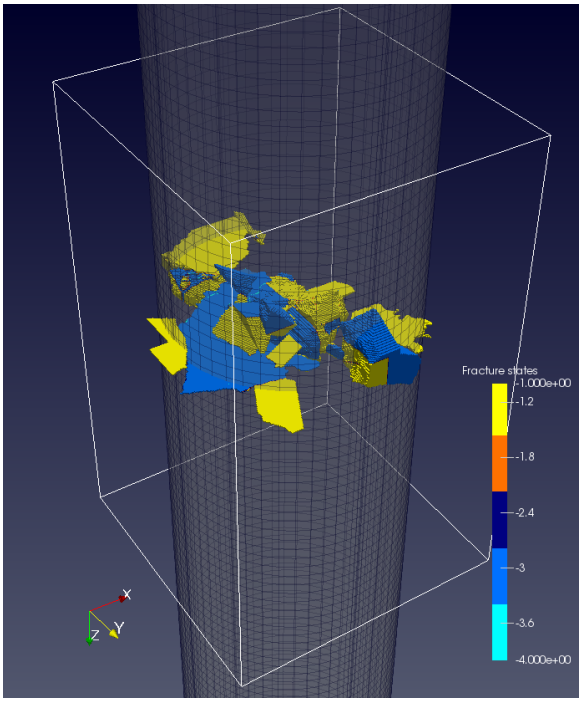

Figure 9. Micro-cracks merging into a macro-crack.

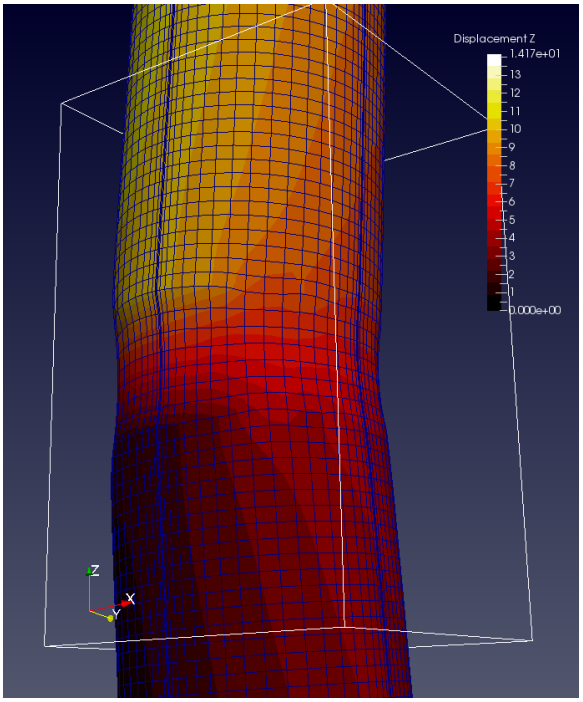

Figure 10. The distorted FE mesh at the end of the CAFE simulation, with the axial displacement contours.

denotes crack edges on 100 planes. -4 (cyan) denotes crack edges on 110 planes.

Fig. 10 shows the FE mesh at the end of the simulation, when the macroscopic cleavage crack has propagated across nearly the whole of the cross section (Fig. 9). The contour plot of the axial displacement is superimposed over the mesh. Note a high displacement gradient across the crack.

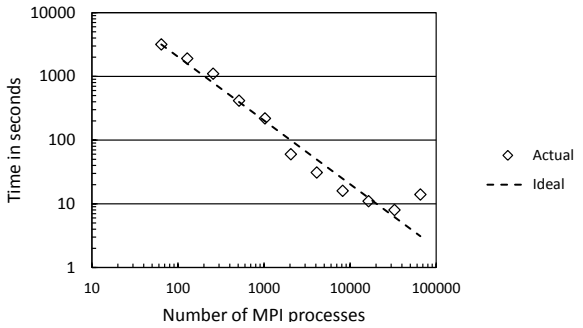

Figure 11. ParaFEM (MPI) scaling for a 3D transient flow explicit analysis on HECToR, Cray XE6. Reproduced from [18].

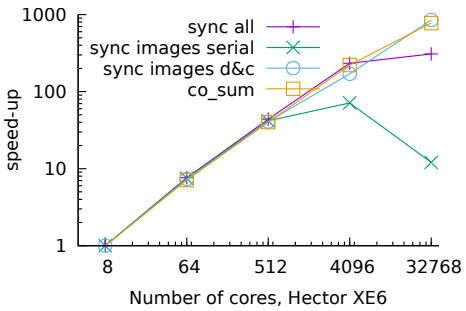

Figure 12. CGPACK (coarrays) scaling for a 3D solidification program on HECToR, Cray XE6. Reproduced from [24].

3. CAFE performance

Individually both ParaFEM and CGPACK libraries showed the potential to scale well into tens of thousands of cores, as seen in Figs. 11 and 12. Fig. 12 shows the effects of different reduction algorithms in a 3D solidification coarray program. It is shown that a coarray collective, `CO_SUM`, gives the most consistent performance up to 32k cores. Note that since both ParaFEM and CGPACK are libraries, scaling analysis makes sense only in context of specific programs built with these libraries.

A representative scaling of a CAFE multi-scale fracture simulation with 1M FE and 800M CA cells on ARCHER, Cray XC30, is shown in Fig. 13. The scaling limit is only about 7k cores (300 Cray XC30 nodes). Prior profiling work of this particular MPI/coarrays program identified several communication and computational hotspots [27], e.g. Fig. 13 shows that replacing an all-to-all communication pattern (`cgca_gcupda` routine) with a nearest neighbour communication (`cgca_gcupdn` routine) increased the scaling limit from 3k to 7k cores.

3.1. CAFE IO

A typical volume of microstructure in a CAFE approach might include 10^6 grains or 10^{11} cells. With 4-byte integers to store cell states, each layer of `space` coarray will take $\approx$ 373GB, i.e. 745GB for both fracture and microstructure datasets. Multi-step CAFE analyses, e.g. progressive fracture

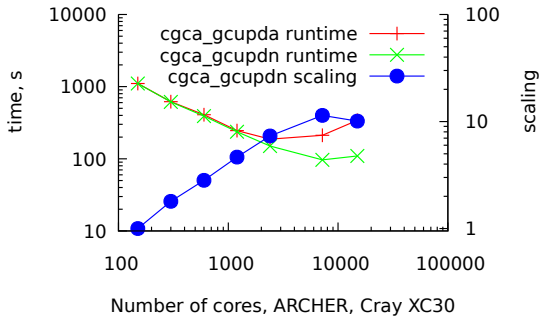

Figure 13. ParaFEM/CGPACK CAFE scaling.

propagation through microstructure, demand that `space` coarray is written to disk at regular intervals. It is clear that efficient coarray IO is required for good scaling.

The Fortran standard does not include parallel IO. However, approaches to achieving high IO performance in MPI programs can be readily applied to coarrays [29], [30].

A single writer/single file serial model is easiest to implement, but has the lowest performance, about 100MB/s on the Cray XE6. In contrast, a multiple writers/single file parallel model has the highest performance using MPI/IO. With some tuning of the Lustre file system, in particular lfs stripe size and stripe count settings, rates of 2.3 GB/s have been achieved. In all cases the CA `space` array coarray is written out as a binary dataset with no metadata. Knowledge of the array extents and of linear spatial resolution is required for post-processing. At present this metadata is written out separately, but work is under way to provide NetCDF [31] and HDF5 IO capability in CGPACK.

At present ParaFEM uses only serial IO. Work is under way to implement MPI/IO in ParaFEM.

4. Synchronising a coarray library

The CGPACK library consists of a number of modules and submodules, with serial and parallel subroutines. A variety of programs can be built, using as many or as few CGPACK library routines as required. The design of the library makes only very basic assumptions on the order of calls to CGPACK routines in a program, e.g. fracture routines must be called after routines establishing microstructure. An error condition is flagged if the order of these routines is reversed. In most other cases the logic of the order of the invocation of CGPACK routines is left to the user.

An immediate consequence of such design is that inter image synchronisation becomes a hard decision. The Fortran standard imposes very strict segment ordering rules to ensure data integrity and to prevent deadlocks [13].

Synchronisation requirements differ for each individual CGPACK routine. For example, a halo exchange algorithm logically maps best onto `SYNC IMAGES` image control statement. Assuming a 3D grid of images, `[:,:,:]`, see Sec. 2.1, each image has to synchronise only with its 26 neighbouring images, i.e. image with the coindex set `[a,b,c]` has to synchronise with images from

[a-1,b-1,c-1] to [a+1,b+1,c+1]. However, the library writer has no way to predict what routines will precede or succeed the halo exchange routine. In practice this often means that the only safe image control statement is SYNC ALL, a global barrier. A fragment of a typical coarray CAFE program might look like this:

```
call cgca_nr( space )  ! sync all inside
call cgca_rt( grt )    ! sync all inside
call cgca_sld( space ) ! sync all inside
call cgca_igb( space )
sync all
call cgca_hxi( space )
sync all
call cgca_gbs( space )
sync all
call cgca_hxi( space )
sync all
sync all
call cgca_gcu( space ) ! local routine
                       ! no sync needed
```

Note that some CGPACK routines include image control statements in the beginning and/or the end, e.g. cgca_sld, the solidification routine and cgca_nr, the nucleation routine. It does not make sense to start cgca_sld on any image until cgca_nr has finished on all images. In such cases the responsibility for arranging sufficient synchronisation has been taken away from the end user. However, in other cases, the user is likely to deploy SYNC ALL to be safe, as shown above.

While excessive use of SYNC ALL might lead to over synchronisation, and hence to poor scaling, our prior profiling analysis on Cray XC30 concluded that the current scaling limit of 3k cores, see Fig. 13, is not related to this.

ParaFEM synchronisation properties are very different, because most of its routines use 2-way message passing MPI calls. In this regard it is very fortunate that in a ParaFEM/CGPACK CAFE program the calls to each library do not alternate often, - there is typically a large chunk of code made of ParaFEM calls, then SYNC ALL, then a large chunk of code made of CGPACK calls, etc. A fragment of a CAFE fracture program is shown below.

```
call cgca_pfem_salloc( nels_pp, nip, nst )
sync all
!end CGPACK part
!start ParaFEM part
CALL rearrange(rest)
elements_0: DO iel=1,nels_pp
 CALL find_g3( g_num_pp(:,iel), &
               g_g_pp(:,iel), rest )
END DO elements_0
```

5. Opportunities for thread parallelisation

Many CA routines contain triple nested loops over all cells on an image. An example below is taken from cgca_clvgp, the cleavage propagation routine. Each iteration of the main loop all cells in the CA on an image are processed.

```
main: do iter = 1,N
 do x3 = lbr(3), ubr(3)
 do x2 = lbr(2), ubr(2)
 do x1 = lbr(1), ubr(1)
  live: if ...
   ! scan only through undamaged cells
   call cgca_clvgn( clvgflag )
   if ( clvgflag ) call sub( space )
  end if live
 end do
 end do
 end do
 call co_sum( clvgglob )
 sync all
 call cgca_hxi( space )
 sync all
 call cgca_dacf( space )
```

Such nested loops might present good opportunities for thread parallelisation with either OpenMP or OpenACC (e.g. on GPUs or Xeon Phi), although the use of underpopulated nodes might be required. Fortran 2008 new intrinsic DO CONCURRENT should also be explored, although at present its performance portability is inferior to OpenMP. Recently the ParaFEM has been ported to Xeon Phi [32]. Porting of CGPACK to Xeon Phi is planned for the future.

6. Conclusions

The design and application of a Fortran coarray CA microstructure simulation library CGPACK has been presented. The use of an integer allocatable array coarray with 4 dimensions and 3 codimensions in CGPACK for a 3D CA polycrystalline microstructure simulation was successful. A microstructure/continuum coarray/MPI deformation and fracture framework has been successfully established by linking together the MPI FE library ParaFEM with CGPACK. Considerable attention has been given to establishing a robust FE to CA mapping data structures and procedures, resulting in a concurrent hierarchical two-way multi-scale CAFE model. Coarrays of derived type with allocatable components were found to be very useful for maintaining dynamic data structures which link the MPI and the coarray parts of the framework. Although some CGPACK coarray CA programs can scale well at least up to 32k cores, at present the fracture CGPACK/ParaFEM multi-scale CAFE model does not scale beyond 3k cores on Cray XC30. A cleavage fracture of a cylindrical ferritic steel specimen was shown as a simple CAFE application. However, a very diverse range of CAFE programs can be created by using ParaFEM with CGPACK. This work proves that interfacing MPI and coarrays is easily achievable. This opens many possibilities for applications in other areas of science and engineering. In addition, because both ParaFEM and CGPACK are distributed under BSD license, the two libraries

can be used to build diverse benchmarking miniapps for compiler and runtime library writers and system architects for exploring parallel performance of coarray and/or coarray/MPI programs.

Acknowledgment

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk). This work was carried out using the computational facilities of the Advanced Computing Research Centre, The University of Bristol (http://www.bris.ac.uk/acrc).

References

[1] W. A. Curtin and R. E. Miller, "Atomistic/continuum coupling in computational materials science," *Model. Simul. Mater. Sci. Eng.*, vol. 11, pp. R33–R68, 2003.

[2] M. Xu and T. Belytschko, "Conservation properties of the bridging domain method for coupled molecular/ continuum dynamics," *Int. J. Numer. Meth. Eng.*, vol. 76, pp. 278–294, 2008.

[3] M. Wallin, W. A. Curtin, M. Ristinmaa, and A. Needleman, "Multi-scale plasticity modeling: Coupled discrete dislocation and continuum crystal plasticity," *J. Mech. Phys. Solids*, vol. 56, pp. 3167–3180, 2008.

[4] M. Rappaz and C. A. Gandin, "Probabilistic modeling of microstructure formation in solidification processes," *Acta Met. Mat.*, vol. 41, pp. 345–360, 1993.

[5] G. Guillemot, C.-A. Gandin, and M. Bellet, "Interaction between single grain solidification and macro segregation: Application of a cellular automaton - finite element model," *J. Crystal Growth*, vol. 303, pp. 58–68, 2007.

[6] C. Zheng and D. Raabe, "Interaction between recrystallization and phase transformation during intercritical annealing in a cold-rolled dual-phase steel: A cellular automaton model," *Acta Materialia*, vol. 61, pp. 5504–5517, 2013.

[7] A. Shterenlikht and L. Margetts, "Three-dimensional cellular automata modelling of cleavage propagation across crystal boundaries in polycrystalline micro- structures," *Proc. Roy. Soc. A*, vol. 471, p. 20150039, 2015.

[8] A. Shterenlikht and I. C. Howard, "The CAFE model of fracture – application to a TMCR steel," *Fatigue Fract. Eng. Mater. Struct.*, vol. 29, pp. 770–787, 2006.

[9] S. Das, A. Shterenlikht, I. C. Howard, and E. J. Palmiere, "A general method for coupling microstructural response with structural performance," *Proc. Roy. Soc. A*, vol. 462, pp. 2085–2096, 2006.

[10] S. J. Wu, C. L. Davis, A. Shterenlikht, and I. C. Howard, "Modeling the ductile-brittle transition behavior in thermomechanically contolled rolled steels," *Met. Mater. Trans. A*, vol. 36, pp. 989–997, 2005.

[11] A. Shterenlikht, S. Margetts, L. McDonald, and N. K. Bourne, "Towards mechanism-based simulation of impact damage using exascale computing," in *Proc. 19th APS Conf. Shock Compression Condensed Matter SCCM-2015, JUN-2015, Tampa, Florida, USA*, 2015.

[12] V. Kouznetsova, W. A. M. Brekelmans, and F. P. T. Baaijens, "An approach to micro-macro modeling of he- terogeneous materials," *Comp. Mech.*, vol. 27, pp. 37–48, 2001.

[13] ISO/IEC 1539-1:2010, *Fortran – Part 1: Base language, International Standard*, 2010.

[14] ISO/IEC JTC1/SC22/WG5 N2074, *TS 18508 Additional Parallel Features in Fortran*, 2015.

[15] A. Shterenlikht, L. Margetts, L. Cebamanos, and D. Henty, "Fortran 2008 coarrays," *ACM Fortran Forum*, vol. 34, pp. 10–30, 2015.

[16] G. Mozdzynski, M. Hamrud, and N. Wedi, "A partitioned global address space implementation of the European centre for medium range weather forecasts integrated forecasting system," *Int. J. High Perf. Comp. Appl.*, vol. 29, pp. 261–273, 2015.

[17] R. Preissl, N. Wichmann, B. Long, J. Shalf, S. Ethier, and A. Koniges, "Multithreaded address space communication techniques for gyrokinetic fusion applications on ultra-scale platforms," in *Supercomputing 2011, Seattle, WA, USA*, 2011.

[18] I. M. Smith, D. V. Griffiths, and L. Margetts, *Programming the Finite Element Method*. Wiley, 5ed, 2014.

[19] I. M. Smith and L. Margetts, "The convergence variability of parallel iterative solvers," *Eng. Computations*, vol. 23, pp. 154–165, 2006.

[20] L. M. Evans, L. Margetts, V. Casalegno, L. M. Lever, J. Bushell, T. Lowe, A. Wallwork, P. Young, A. Lindemann, M. Schmidt, and P. M. Mummery, "Transient thermal finite element analysis of CFC-Cu ITER monoblock using X-ray tomography data," *Fusion Eng. Des.*, vol. 100, pp. 100–111, 2015.

[21] J. D. Arregui-Mena, L. Margetts, D. V. Griffiths, L. Lever, G. Hall, and P. M. Mummery, "Spatial variability in the coefficient of thermal expansion induces pre-service stresses in computer models of virgin gilsocarbon bricks," *J. Nuclear Mater.*, vol. 465, pp. 793–804, 2015.

[22] F. Levrero-Florencio, L. Margetts, E. Sales, S. Xie, K. Manda, and P. Pankaj, "Evaluating the macroscopic yield behaviour of trabecular bone using a nonlinear homogenisation approach," *J. Mech. Behavior Biomed. Mater.*, vol. 61, pp. 384–96, 2016.

[23] S. D. Rawson, L. Margetts, J. K. F. Wong, and S. H. Cartmell, "Sutured tendon repair; a multi-scale finite element model," *Biomechanics Modelling Mechanobiology*, vol. 14, pp. 123–133, 2015.

[24] A. Shterenlikht, "Fortran coarray library for 3D cellular automata microstructure simulation," in *Proc. 7th PGAS Conf., 3-4 October 2013, Edinburgh, Scotland, UK*, M. Weiland, A. Jackson, and N. Johnson, Eds. The University of Edinburgh, 2014, pp. 16–24. [Online]. Available: http://www.pgas2013.org.uk

[25] A. Fanfarillo, "Parallel programming techniques for heterogeneous exascale computing platforms," Ph.D. dissertation, University of Rome Tor Vergata, 2016. [Online]. Available: http://www.opencoarrays.org/uploads/6/9/7/4/69747895/fanfarillophd.pdf

[26] A. Fanfarillo, T. Burnus, S. Filippone, V. Cardellini, D. Nagle, and D. Rouson, "OpenCoarrays: open-source transport layers supporting coarray Fortran compilers," in *Proc. PGAS 2014, Eugene, Oregon, USA*, 2014. [Online]. Available: http://www.opencoarrays.org/uploads/6/9/7/4/69747895/pgas14_submission_7-2.pdf

[27] L. Cebamanos, A. Shterenlikht, D. Arregui-Mena, and L. Margetts, "Scaling hybid coarray/mpi miniapps on archer," in *Cray User Group 2016 meeting (CUG2016), London, 8-12-MAY-2016*, 2016.

[28] J. Phillips, A. Shterenlikht, and M. J. Pavier, "Cellular automata modelling of nano-crystalline instability," in *Proc. 20th UK ACME Conf. 27-28 March 2012, The University of Manchester, UK*, 2012.

[29] D. Henty, A. Jackson, C. Moulinec, and V. Szeremi, " Performance of Parallel IO on ARCHER, version 1.1," ARCHER White Papers, 2015. [Online]. Available: http://archer.ac.uk/documentation/white-papers/parallelIO/ARCHER_wp_parallelIO.pdf

[30] Y. L. Zou, W. Xue, and S. S. Liu, "A case study of large-scale parallel I/O analysis and optimization for numerical weather prediction system," *Future Gen. Comp. Systems*, vol. 37, pp. 378–389, 2014.

[31] T. Collins, "Using NetCDF with Fortran on ARCHER, version 1.1," ARCHER White Papers, 2016. [Online]. Available: http://archer.ac.uk/documentation/white-papers/fortanIO_netCDF/fortranIO_netCDF.pdf

[32] L. Margetts, J. D. Arregui Mena, T. Hewitt, and L. Mason, "Parallel finite element analysis using the intel xeon phi," in *Proc. Emerging Technology Conf. (EMiT 2016), ISBN 978-0-9933426-3-9*, 2016.